# Exam BIO134: Programming in Biology HS2020

## 15.01.2021; 10:30-13:00

General comments:

* I needed approx. 113 min for all exercises (with opening files and reading questions, but without writing my comments and without bathroom break). The first three were very quick, correct solution at the first attempt. For question 4 I took a overly complicated route by implementing my own sorting algorithm, which took time, but when I realized it worked with sort(), question 4 was much faster (about 5-6 min if I had the right idea immediately). 5, 6 and 7 were generally longer, due to the data structures being less straight-forward and just more things to do. The exam is thus for sure doable in 150 min, at least for better students. However, as 30-40 min of buffer is not too much, it is very likely that many students won’t be able to finish all exercises.
* As mentioned in the comments, I think putting all variables mentioned in the texts in Courier New, the same font as the code. This makes it clearer when you refer to a variable, makes the distinction clearer between variables and the file name (can stay in italicized Calibri), and is also usually how variables are written in text books.
* I corrected some words in the texts which made the text for me easier to understand. This is of course objective and does not mean that these changes should 100% be kept (except the few typos and missing dots :D )
* Usually the question texts say that “the program should be general enough that it would still work with different list/string/etc...”. In the course they always had one “training” variable that they could use to write their program, and then a “test” variable (usually a longer list/string) that they could use to test if their program is general enough. Here you mention the importance of generality, but do not provide a test variable. Is this on purpose, and might a test variable (together with expected output) be useful for them to see if their program is general enough?

## Question 1

text = 'tgovoyd vlwucqk'

numbers = [3, 9, 7, 0, 9, 5, 7, 8, 2, 8, 3, 6, 7, 0, 6]

Write a program that creates a new string based on the string textand the list numbers. The new string should contain the letters of text at those positions where the integers in the list numbers are larger than 5, in the order they occur in text. It should then print the string:

good luck

The program should be general enough that it would still work according to the same principle, if the list numbers and the string text would be of different length and/or contain different values. You may assume that the list numbers always contains positive integers, and that it always has the same length as the string text.

*Time required to solve: 2 min*

## Question 2

rna = 'AUGUUCGAA'

Write a program that, based on the string rna, creates a dictionary, where the keys are the bases A, U, G, C and the values indicate the positions where the respective bases occur in the sequence rna.

The program should then print the dictionary:

{'A': [0, 7, 8], 'U': [1, 3, 4], 'G': [2, 6], 'C': [5]}

Note that a dictionary does not have a specified order, so that the same dictionary can be printed in many different ways. It is only important that every key has the correct associated value.

The program should be general enough that it would still work with another sequence of different length.

*Time required to solve: 2 min*

## Question 3

import numpy as np

np.random.seed(0)

Start your code with the lines above. Then, write a **function** n\_times\_to\_threshold() that takes two positive integer numbers as input: a threshold and a maximum number. Based on these two input variables, the function should sum up random numbers ranging from 1 to (and including) the maximum number until the sum of all the random numbers is larger than the threshold. The random numbers should be generated within the function by using np.random.randint(1, max\_number+1).

The function should return the number of times a random number was generated until the sum exceeded the threshold.

Call the function and print the result using 43 as the threshold and 10 as the maximum number by using the following code:

k = n\_times\_to\_threshold(43, 10)

print('The '+str(k)+'th number has brought the sum above the threshold!')

For the given numbers it should then print:

The 9th number has brought the sum above the threshold!

*Time required to solve: 3.5 min*

## Question 4

Here is a list of our solar system’s planets (name, diameter [km], distance from the sun [km]) in alphabetical order:

planets = [['Earth', 12742, 149598262], ['Jupiter', 139822, 778340821], ['Mars', 6779, 227943824], ['Mercury', 4878, 57909227],

['Neptune', 49244, 4498396441], ['Saturn', 116464, 1426666422],

['Uranus', 50724, 2870658186], ['Venus', 12104, 108209475]]

Write a program that sorts the planets according to their distance from the sun, and prints, based on the sorted list, the exact following lines:

Mercury .. 4878 km in diameter 57909227 km away from the sun

Venus .... 12104 .............. 108209475 ....................

Earth .... 12742 .............. 149598262 ....................

Mars ..... 6779 .............. 227943824 ....................

Jupiter .. 139822 .............. 778340821 ....................

Saturn ... 116464 .............. 1426666422 ....................

Uranus ... 50724 .............. 2870658186 ....................

Neptune .. 49244 .............. 4498396441 ....................

Your program should still work if the list planets contained information about only a subset of the planets (some entries in planets removed).

*Time required to solve: first 19 min (implementing bubble sort), then 5 min (using sort() and finding distances in planets) on second try*

## Question 5

The file *‘fMRI\_series.txt’* contains two numbers on each line, separated by spaces. The first number is the timepoint in seconds and the second number is the activity level in arbitrary units at that given timepoint measured by fMR imaging of a mouse brain.

Starting at timepoint 170s, an external stimulus was presented to the mouse every 60 seconds for 5 times, while continuously recording the activity level every 5 seconds.

Write a program that extracts the data from the file and creates a numpy array data with the activity levels as floats, starting with the activity level corresponding to timepoint 170s. The array should have 5 rows (one row per round of stimulation), and 12 columns (one column per timepoint in 60 seconds). Calculate and print the mean of the 5 stimulation rounds for each timepoint during the 60 second period using the following code:

import numpy as np

# Create “data” somewhere in your program, then:

print(np.mean(data, 0))

It should then print the following array:

[13622.021218 13924.207528 13901.032028 13897.297232 13978.789918

13937.638388 13888.950176 13874.45905 13711.046962 13664.24686

13557.45596 13617.378826]

The code should be general enough that it would still work according to the same principle if *fMRI\_series.txt* contained a dataset with a different number of stimulation rounds. You may assume that activity levels are always recorded every 5 seconds, stimulations are always 60s apart, and that the first stimulation always starts at the timepoint 170s.

*Time required to solve: 26 min*

## Question 6

aminoacids=['alanine','cysteine','aspartic acid','glutamic acid', 'phenylalanine','glycine','histidine','isoleucine','lysine', 'leucine','methionine','asparagine','proline','glutamine', 'arginine','serine','threonine','valine','tryptophan','tyrosine']

The list aminoacids contains the 20 standard amino acids. In addition to their full name, there is a standard one letter code, that uniquely identifies the amino acids.

Write a program to create your own one letter code. Use the following rules for this:

1. If the first letter of the name is unique, use it as a one letter code (eg. P for Proline)
2. For the other amino acids that share the first letter:
   1. Assign the first letter as code to the amino acid with the shortest name in the group of amino acids with the same first letter (eg. ['Glycine’, Glutamic acid', 'Glutamine']: G for Glycine). You may assume that there is exactly one shortest name per group.
   2. For the rest: Approach the other amino acids with the same first letter in alphabetical order and choose the next free letter in the alphabet as the one letter code.

The program should print your one letter code in the following manner (note the capital letter in the full names):

A Alanine

B Arginine

C Cysteine

D Asparagine

E Aspartic acid

F Glutamic acid

G Glycine

H Histidine

I Isoleucine

J Glutamine

K Leucine

L Lysine

M Methionine

N Phenylalanine

O Threonine

P Proline

Q Tryptophan

R ---

S Serine

T Tyrosine

U ---

V Valine

W ---

X ---

Y ---

Z ---

Your program should still work if amino acids were named differently or if there was a different number of amino acids. You may assume that there are not more than 26 amino acids.

Please note, that the points you get for this question are largely based on your printed output and you will be given 0 points if the program prints nothing. In case not all the printed values are correct, you may still get part of the points.

*Time required to solve: 32.5 min*

## Question 7

functional\_motifs = ['GAGGTAAAC','TCCGTAAGT','AAGGTTGGA','ACAGTCAGT','TAGGTCATT','TAGGTACTG','ATGGTAACT','CAGGTATAC','TGTGTGAGT','AAGGTAAGT']

query = 'ACTCAGCCCCAGCGGAGGTGAAGGACGTCCTTCCCCAGGAGCCGGTGAGAAGCGCAGTCGGGGGCACGGGGATGAGCTCAGGGGCCTCTAGAAAGATGTAGCTGGGACCTCGGGAAGCCCTGGCCTCCAGGTAGTCTCAGGAGAGCTACTCAGGGTCGGGCTTGGGGAGAGGAGGAGCGGGGGTGAGGCCAGCAGCA'

Assume you are interested in finding a transcription factor binding site in a given DNA sequence. The transcription factor binding site is not a strictly defined sequence, but there are some preferred characteristics in a 9 base long motif. You know the sequences of 10 exemplary binding sites for this transcription factor (list functional\_motifs).

A profile, also called a position specific scoring matrix, is a motif descriptor that tries to capture the intrinsic

variability characteristic of sequence patterns. Such a profile has 4 rows, one for each of the bases A, C, G and T, and as many columns as the sequence motif length. The profile for the given 10 sequences is as follows: each number in the matrix indicates the frequency with which a given nucleotide has been observed at a given position.

0 1 2 3 4 5 6 7 8

A 0.4 0.6 0.1 0.0 0.0 0.6 0.7 0.2 0.1

C 0.1 0.2 0.1 0.0 0.0 0.2 0.1 0.1 0.2

G 0.1 0.1 0.7 1.0 0.0 0.1 0.1 0.5 0.1

T 0.4 0.1 0.1 0.0 1.0 0.1 0.1 0.2 0.6

Write a program to create and print the profile matrix yourself from the 10 given sequences. The data type you use for this matrix (eg. numpy array, dictionary, list) is up to you. The program should print this matrix such that all frequencies are visible, but the precise formatting of the matrix is not relevant (simply print your variable).

Then, apply the profile matrix to the sequence query and find all the potential binding sites. For this, create a score for each possible motif of length 9 by adding up the frequencies for the occurring nucleotides of the current motif.

Print the motifs with a score larger than 4.4:

position 14: GAGGTGAAG, 4.5

position 127: CAGGTAGTC, 4.5

Next, generate a hypothetical, ideal motif that would have the maximum score, and print it (In this particular example there are two possible ideal motifs. It is enough to find one).

AAGGTAAGT or TAGGTAAGT

Your program should still work with a different motif and query sequence.

Please note, that the points you get for this question are largely based on your printed output and you will be given 0 points if the program prints nothing. In case not all the printed values are correct, you may still get part of the points.

*Time required to solve: 28 min*